**Державний вищий навчальний заклад**

**Ужгородський національний університет**

**Факультет інформаційних технологій**

**Лабораторна робота №6**

**Алгоритми на деревах**

Виконав студент 1 курсу

спеціальності “Інженерія

програмного забезпечення”

Дєлов В’ячеслав Едуардович

**Ужгород-2025**

**Мета:** набути навичок створення та обробки дерев.

**Завдання до практичної роботи**

1. Написати код програми відповідно до обраного варіанту. Кожен варіант містить завдання та спосіб обходу дерева для кожного із завдань.
2. Оформити звіт та завантажити його в системі електронного навчання ДВНЗ «УжНУ» в установлений термін.
3. Підготувати відповіді на контрольні питання.

**Варіант 18 (9)**

**Завдання 1**

Метод обходу: Симетричний

Ті елементи дерева, які складаються з однакових цифр, замінити значенням “0”.

**Хід роботи:**

Код програми:

class Node {

  constructor(value) {

      this.value = value;

      this.left = null;

      this.right = null;

  }

}

class BinarySearchTree {

  constructor() {

      this.root = null;

  }

  insert(value) {

      const newNode = new Node(value);

      if (!this.root) {

          this.root = newNode;

          return;

      }

      let current = this.root;

      while (true) {

          if (value < current.value) {

              if (!current.left) {

                  current.left = newNode;

                  return;

              }

              current = current.left;

          } else {

              if (!current.right) {

                  current.right = newNode;

                  return;

              }

              current = current.right;

          }

      }

  }

  // перевірка однакових цифр

  hasSameDigits(number) {

      const str = number.toString();

      if (str.length === 1) return false;

      let arr = str.split('');

      return arr.every(ch => ch === str[0]);

  }

  // заміна значень

  inOrderReplace(node = this.root) {

      if (!node) return;

      this.inOrderReplace(node.left);

      if (this.hasSameDigits(node.value)) {

          node.value = 0;

      }

      this.inOrderReplace(node.right);

  }

  inOrderPrint(node = this.root) {

      if (!node) return;

      this.inOrderPrint(node.left);

      console.log(node.value);

      this.inOrderPrint(node.right);

  }

}

// Тест

const tree = new BinarySearchTree();

tree.insert(111);

tree.insert(5);

tree.insert(2222);

tree.insert(78);

tree.insert(33);

tree.insert(444);

tree.insert(9);

console.log('До заміни:');

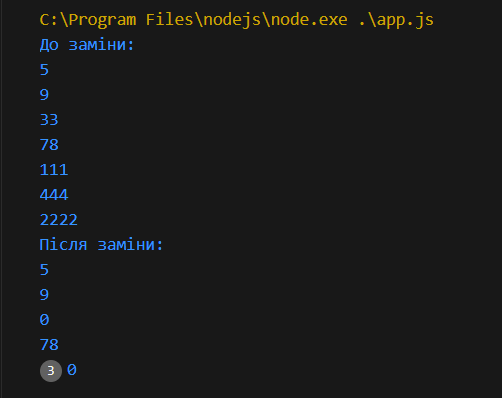
tree.inOrderPrint();

tree.inOrderReplace();

console.log('Після заміни:');

tree.inOrderPrint();

Результат:



**Завдання 2**

Метод обходу: Симетричний

Вивести на екран ті вузли, у яких різниця сум елементів в правому і лівому піддереві більша 20. Вивести ці суми для кожного вузла

Код програми:

class Node {

    constructor(value) {

        this.value = value;

        this.left = null;

        this.right = null;

        this.subtreeSum = 0;

    }

}

class BinarySearchTree {

    constructor() {

        this.root = null;

    }

    insert(value) {

        const newNode = new Node(value);

        if (!this.root) {

            this.root = newNode;

            return;

        }

        let current = this.root;

        while (true) {

            if (value < current.value) {

                if (!current.left) {

                    current.left = newNode;

                    return;

                }

                current = current.left;

            } else {

                if (!current.right) {

                    current.right = newNode;

                    return;

                }

                current = current.right;

            }

        }

    }

    // всі суми піддерев

    calculateSubtreeSums(node = this.root) {

        if (!node) return 0;

        const leftSum = this.calculateSubtreeSums(node.left);

        const rightSum = this.calculateSubtreeSums(node.right);

        node.subtreeSum = node.value + leftSum + rightSum;

        return node.subtreeSum;

    }

    inOrderCheck(node = this.root) {

        if (!node) return;

        this.inOrderCheck(node.left);

        const leftSum = node.left ? node.left.subtreeSum : 0;

        const rightSum = node.right ? node.right.subtreeSum : 0;

        const diff = Math.abs(leftSum - rightSum);

        if (diff > 20) {

            console.log(`Вузол ${node.value}: ліва сума = ${leftSum}, права сума = ${rightSum}, різниця = ${diff}`);

        }

        this.inOrderCheck(node.right);

    }

}

// Тест

const tree = new BinarySearchTree();

tree.insert(50);

tree.insert(30);

tree.insert(70);

tree.insert(20);

tree.insert(40);

tree.insert(60);

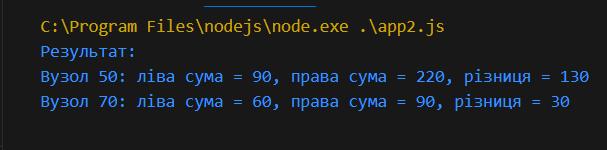
tree.insert(90);

console.log("Результат:");

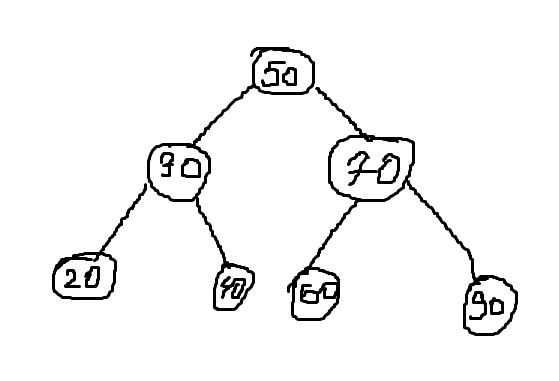
tree.calculateSubtreeSums();

tree.inOrderCheck();

Результат:



Перевірка:

Вигляд дерева: 

Для вузла 30: ліве піддерево = 20, праве = 40, різниця 20 не > 20  
Для вузла 50: ліве піддерево = 90, праве = 220, різниця 130 > 20  
Для вузла 70: ліве піддерево = 60, праве = 90, різниця 30 > 20

Код програми запушено на гіт-хаб в репозиторій https://github.com/DelovSlava/Algoritmi\_-\_Strutures

**Висновок:**

У ході виконання завдання було реалізовано симетричний (in-order) обхід бінарного дерева.

* реалізовано завдання із заміною значення вузлів, які складаються з однакових цифр, на 0. Перевірка виконувалась під час симетричного обходу дерева.
* Знайдено та виведено ті вузли, у яких різниця сум елементів правого і лівого піддерев перевищує 20. Для оптимізації було попередньо обчислено суму піддерева кожного вузла, що дозволило зменшити кількість обчислень і підвищити ефективність програми до O(n) замість O(n^2)

Робота демонструє ефективне використання рекурсії, оптимізацію підрахунків та правильне застосування властивостей бінарного дерева.